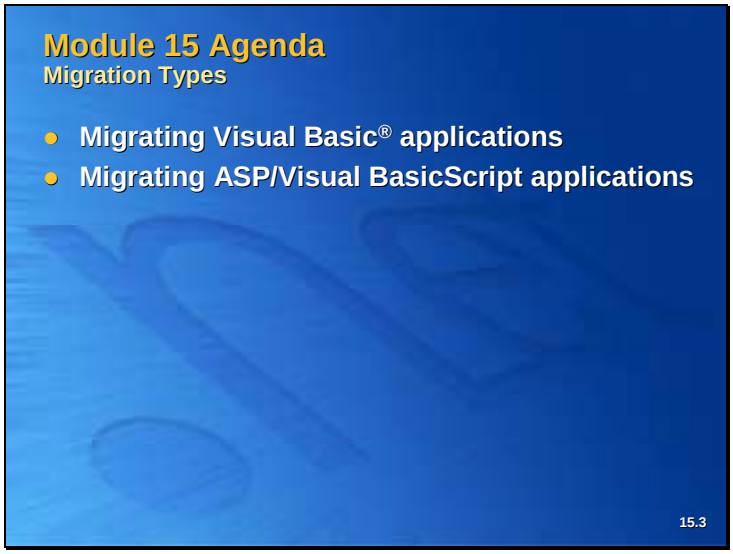
Microsoft
.net developer
training

Migrating Applications to .NET

Agenda: Day 2

- Module 09: Creating XML Web Services
- Module 10: Code Instrumentation and Middle Tier Technology
- Module 11: Security in .NET
- Module 12: Lifecycle Tools for .NET
- Module 13: Configuring and Deploying .NET Applications
- Module 14: Application Interoperability
- ▶ Module 15: Migrating Applications to .NET
- Module 16: Building Applications for Mobile Devices

 ©2002 Microsoft Corporation

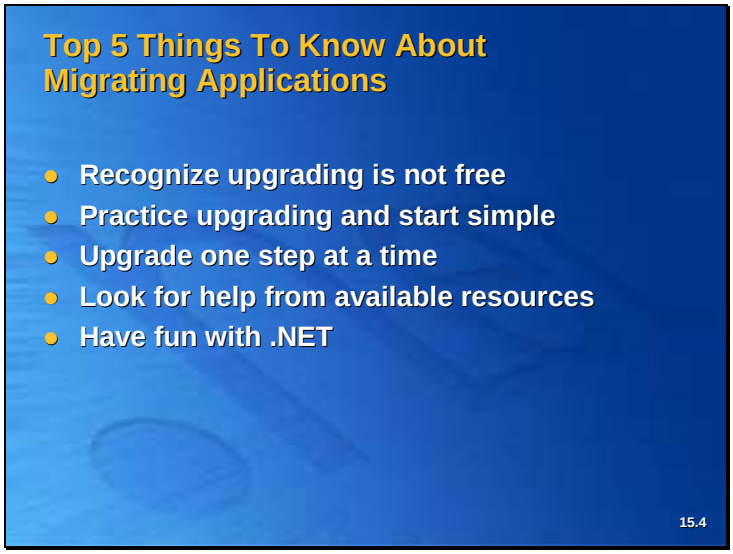


Module 15 Agenda

Migration Types

What you will learn:

- Steps involved in migrating your applications to .NET
- How to plan for migration and select applications that will migrate to .NET
- How to proceed with your migration, what the tools can do for you and what you will have to do
- What fails to migrate


Top 5 Things To Know About Migrating Applications
Recognize upgrading is not free
Practice upgrading and start simple
Upgrade one step at a time
Look for help from available resources
Have fun with .NET
15.4

Top 5 Things To Know About Migrating Applications

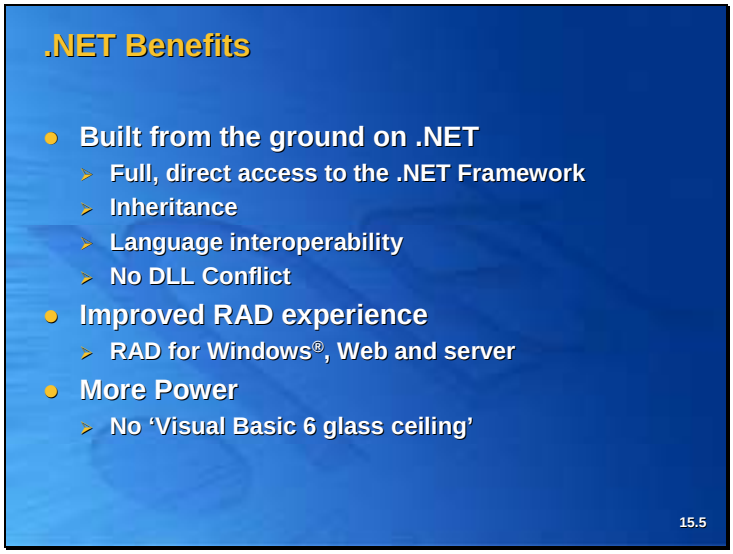
1. **Recognize upgrading is not free:** Visual Basic .NET introduces a host of powerful new concepts, which demand that even seasoned developers learn the product and how to upgrade.
2. **Practice upgrading and start simple:** Start with a small application (one form). Visual Basic .NET Beta 2 is great for upgrading modules, classes, forms and ADO. Unless you're experienced, avoid projects with areas that have no automatic upgrade path - DAO/RDO databinding; ActiveX DHTML pages; Add-ins; memory tweaking functions; user controls and Web classes. Support for these will be added in the shipping version.
3. **Upgrade one step at a time:** For large applications, upgrade one component at a time and work your way up the dependency tree. For example, upgrade the client application before upgrading the middle tier. Get each component working before moving to the next.
4. **Look for help:** Microsoft and its partners are releasing information on how to upgrade, common problems and what cool new features to add. This info will answer most of your questions, but for starters, here are three important resources:

   A Webcast talk on upgrading:
   http://support.microsoft.com/servicedesks/webcasts/wc022701/wcblurb022701.asp

   How to prepare your Visual Basic 6 application for the upgrade to Visual Basic .NET:
   http://msdn.microsoft.com/library/techart/Visual Basic 6tovbdotnet.htm

   Transition from Visual Basic to Visual Basic .NET:
   http://msdn.microsoft.com/vbasic/technical/upgrade/default.asp
5. **Have fun with .NET:** Upgrading an application is quicker than rewriting it. Your upgraded code is the foundation for your .NET development and once you learn a few basic principles, it is no longer a chore. It's like learning to ski. It takes time to get on your feet, but then you're hooked.

 ©2002 Microsoft Corporation



.NET Benefits

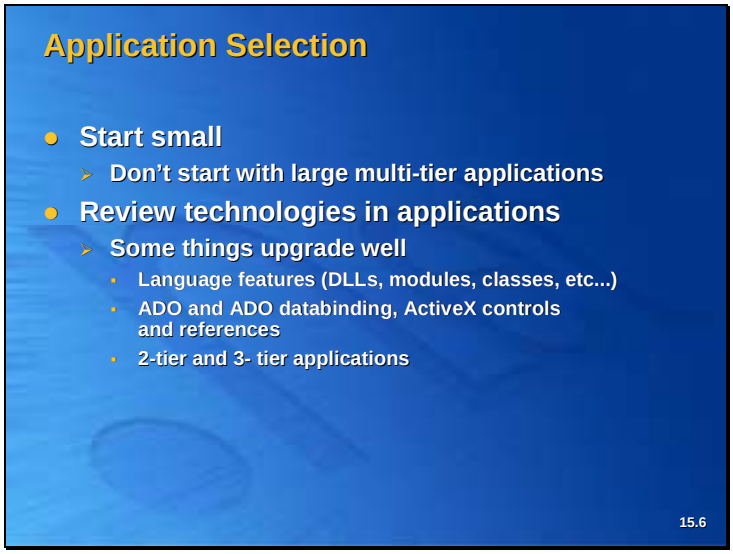
Any computer that runs managed code requires the Common Language Runtime (CLR) and the .NET Framework libraries. This means that you will need to deploy the CLR to each Web server and middle-tier server that runs managed code.

Starting with Beta 2 of the .NET Framework, Microsoft includes with the framework a redistributable package for the .NET runtime files. The redistribution package contains the core CLR engine as well as .NET Framework pieces and ASP.NET. This package can be used to deploy the runtime and framework to each computer in your application. The redistribution package is a standard Microsoft Installer (MSI) package.

Assemblies:
All .NET components are packaged into assemblies. An assembly, typically packaged as one or more DLLs, is a collection of resources and .NET components together with the metadata necessary to describe the components.

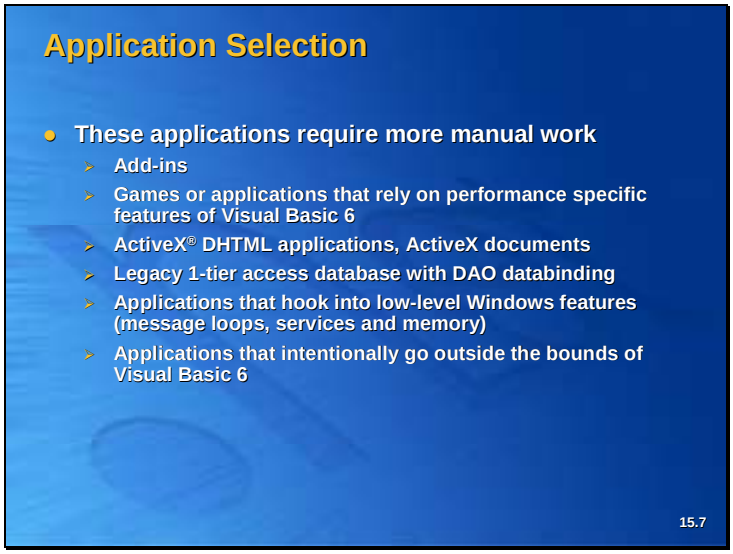
Unlike COM objects, which are located by means of an entry in the Windows registry, .NET components are found based on the location of their assembly in the file system. Additionally, COM components and component registration are shared across the computer.

In contrast, each .NET application can have a private copy of a .NET component with different applications able to have a different version of the same component deployed for that application.


Application Selection
Start small
Don't start with large multi-tier applications
Review technologies in applications
Some things upgrade well
Language features (DLLs, modules, classes, etc...)
ADO and ADO databinding, ActiveX controls and references
2-tier and 3- tier applications
15.6

Application Selection

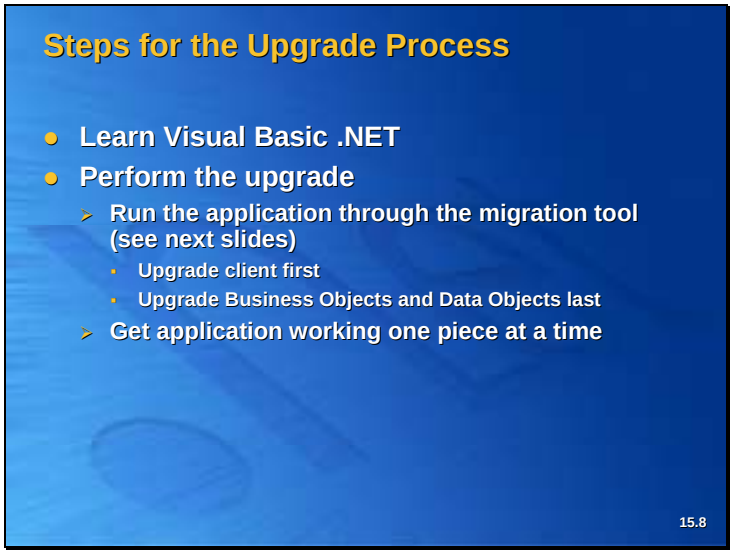
 ©2002 Microsoft Corporation


Application Selection
These applications require more manual work
Add-ins
Games or applications that rely on performance specific features of Visual Basic 6
ActiveX® DHTML applications, ActiveX documents
Legacy 1-tier access database with DAO databinding
Applications that hook into low-level Windows features (message loops, services and memory)
Applications that intentionally go outside the bounds of Visual Basic 6
15.7

Application Selection

The following application types require you to manually upgrade them to .NET:

- ActiveX documents
- DHTML ActiveX designer pages
- DAO and RDO databinding
- OLE, shape controls
- Vertical lines
- Property pages
- Add-in extensibility
- Language features
- LSet, ObjPtr, VarPtr, StrPtr
- GoSub and Return, Computed GoTo


Steps for the Upgrade Process
Learn Visual Basic .NET
Perform the upgrade
Run the application through the migration tool (see next slides)
Upgrade client first
Upgrade Business Objects and Data Objects last
Get application working one piece at a time
15.8

Steps for the Upgrade Process

1. Learn Visual Basic .NET because the more you know, the easier it will be. Upgrading is an inappropriate way to learn and debugging is not as good. Upgrading requires a different set of skills than developing.

   Prepare your application. Refer to the white paper (link):

   http://msdn.microsoft.com/library/techart/Visual Basic 6tovbdotnet.htm

   Also refer to the Web cast
   http://support.microsoft.com/servicedesks/webcasts/wc022701/wcblurb022701.asp

2. Perform the upgrade.

   a. Run the existing application through the migration tool. The tool only does VB projects. If you've got a project group or n-tier application, upgrade the client/downstream dependencies first.

   b. Get the application working one piece at a time

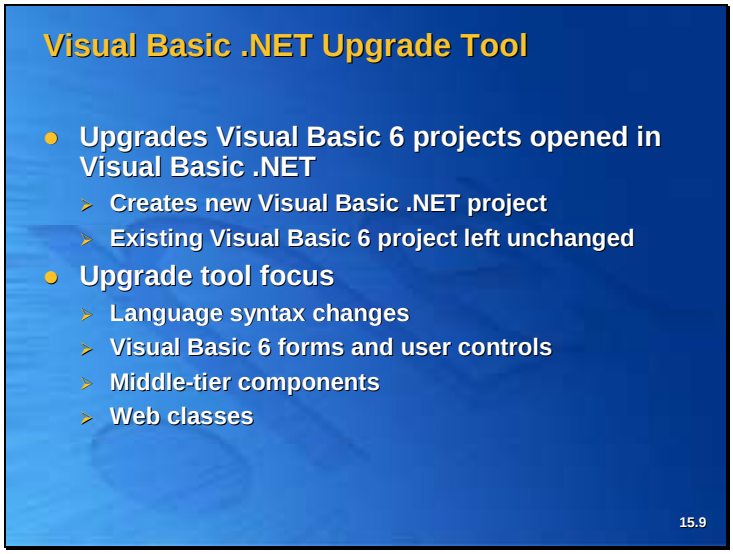
      Consider a VB 6.0 application that consists of a VB 6.0 Client and VB 6.0 Business Objects. In such an application the following steps are recommended for a successful upgrade:

      1. Upgrade the VB 6.0 client to the .NET client. At this point the .NET client will still be accessing the VB 6.0 Business Objects. Get this working first.
      2. Next, upgrade the VB 6.0 Business Objects to .NET Business Objects. The client still references the existing VB 6.0 Business Objects.
      3. Finally, change the reference from the. NET client to the .NET Business Objects.

The idea is to keep the application working at the end of each step. Upgrade a bit and get it working, then upgrade the second bit and get it working.

Why this order? Using COM from .NET is easier than accessing .NET from COM, plus you don't have to change the Visual Basic 6. You upgrade it and then change it, always working with the new stuff.

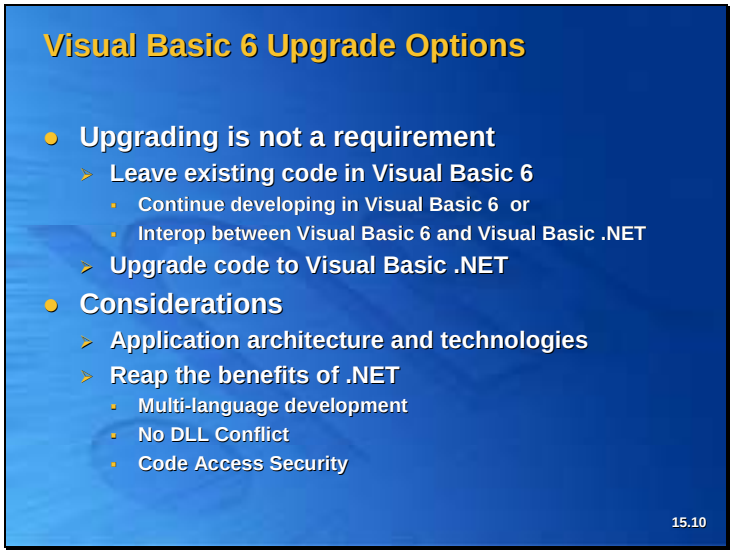
 ©2002 Microsoft Corporation


Visual Basic .NET Upgrade Tool
Upgrades Visual Basic 6 projects opened in Visual Basic .NET
Creates new Visual Basic .NET project
Existing Visual Basic 6 project left unchanged
Upgrade tool focus
Language syntax changes
Visual Basic 6 forms and user controls
Middle-tier components
Web classes
15.9

Visual Basic .NET Upgrade Tool

Upgrade tool action

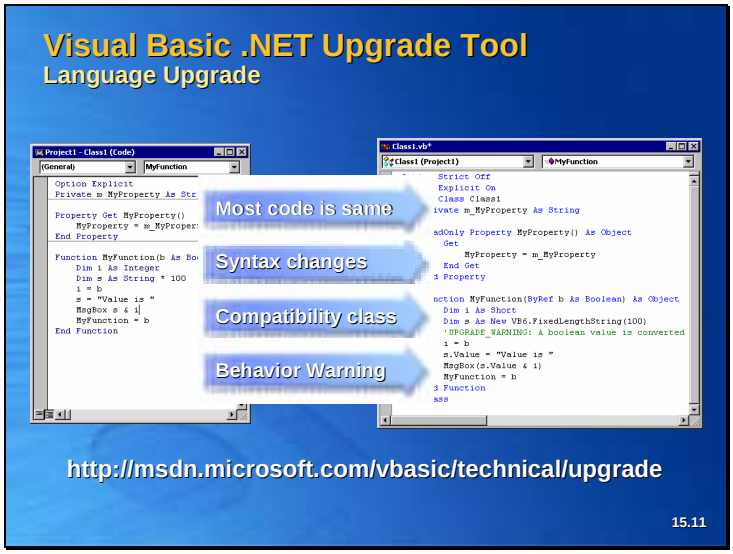
- Where possible: Upgrade to new objects
- If necessary: Compatibility classes
- Where different: Put TODO comment in code


Visual Basic 6 Upgrade Options
Upgrading is not a requirement
Leave existing code in Visual Basic 6
Continue developing in Visual Basic 6 or
Interop between Visual Basic 6 and Visual Basic .NET
Upgrade code to Visual Basic .NET
Considerations
Application architecture and technologies
Reap the benefits of .NET
Multi-language development
No DLL Conflict
Code Access Security
15.10

Visual Basic 6 Upgrade Options

The first thing to consider in terms of a migration is whether to migrate the code at all. The COM interoperability features of the .NET Framework are powerful and, in nearly all cases, they allow you to continue to use your existing code without migrating it to managed code. As you develop new parts of your application or reuse components of your application from newer managed code applications, in most cases, you can call your existing components through the COM interoperability functionality provided by .NET.

 ©2002 Microsoft Corporation



Visual Basic .NET Upgrade Tool

Language Upgrade

After upgrading, the first thing you will notice is that your code remains similar to Visual Basic.

There are some differences, however, such as the more explicit upgraded code, the tool resolves default properties, adds ByRef to procedure parameters and declares implicitly declared variables. The result is more explicit and readable code.

The **tool** performs any syntax changes:

- The property Get/Let syntax has changed
- Integer is replaced with short
- Parentheses are added to functions

If there is a behavioral change, the upgrade tool will mark your code with a warning. In this example, a Boolean is being coerced to an Integer, so the upgrade tool inserts a warning. These warnings show up in the task list and each are linked to online help. Our documentation team has done a tremendous job writing explanations for each issue and telling you what to do next.

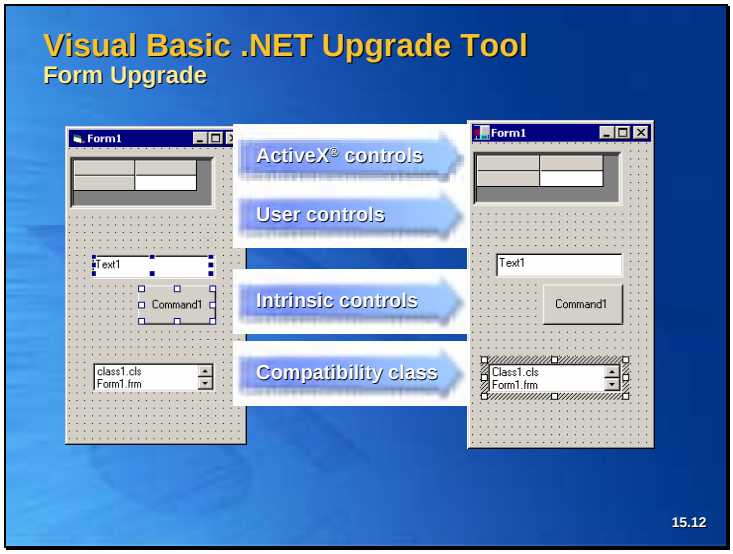
Most code is the same - Visual Basic .NET is Visual Basic so most of your constructs will remain the same (conditionals like if-then, procedures, etc.) You will immediately recognize the upgraded code.

Where necessary, syntax changes will be made to the language:

- Visual Basic 6 integers are now shorts
- Variants are now Objects
- Parameter-less default properties are resolved

When this is impossible, compatibility classes are used to get equivalent Visual Basic 6 functionality in Visual Basic .NET. These allow you to run code, but it may be at a performance hit.

If something is unable to be upgraded, a warning is inserted in your code.



Visual Basic .NET Upgrade Tool

Form Upgrade

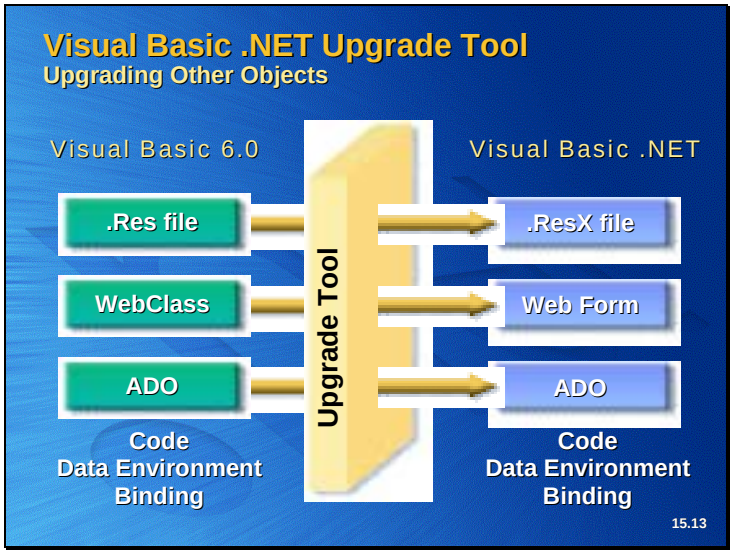
After upgrading is complete, your forms will look the same.

Windows Forms is a great container for Microsoft and Third party ActiveX controls.

Text boxes, command buttons, intrinsic controls and user controls are upgraded.

Windows Forms does not have a FileListBox control, or control arrays, so the Visual Basic team has created various extender controls that give exactly the same behavior as Visual Basic 6. These are in the Visual Basic 6 compatibility library.

 ©2002 Microsoft Corporation

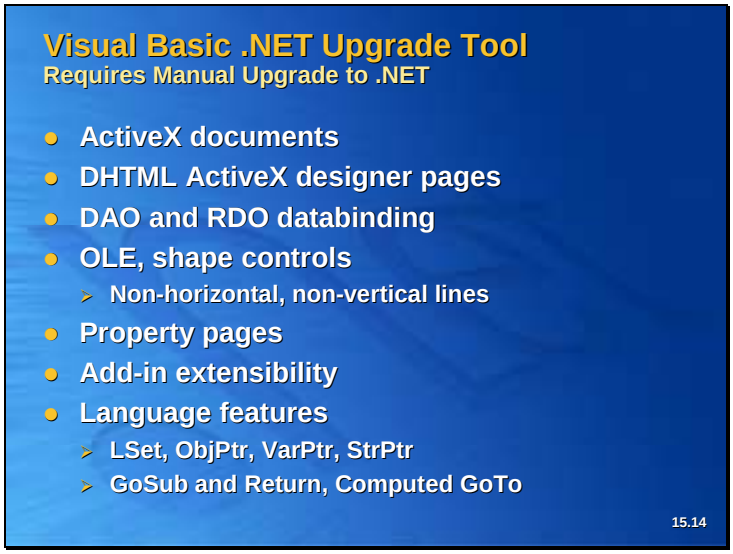


Visual Basic .NET Upgrade Tool

Upgrading Other Objects

To summarize other technologies in your Vb6 project:

- **Resource files** are upgraded to .NET ResX files, which can store any .NET type
- **Web Classes → Web Forms**
- The **ADO data environment** is upgraded to a class that has the same behavior
- **Code and databinding** are supported in .Net just like in Visual Basic 6



Visual Basic .NET Upgrade Tool

Requires Manual Upgrade to .NET

There are some technologies that the tool cannot upgrade automatically.

ActiveX Documents & ActiveX DHTML Designer pages are a result of bolting Web features onto Visual Basic 5 and Visual Basic 6. They generate Fat clients where code is downloaded onto the client machine, only work on Windows and they require installation of the Visual Basic 6 runtime.

In Visual Basic .NET, the Web story has been re-architected to be scalable, secure and cross platform with a true thin client. The processing is done at the server.

The different style of the new architecture makes an automatic upgrade impossible. You should leave these applications in Visual Basic 6. They are not redundant. You can navigate from a Web Form to an ActiveX doc to ActiveX DHTML pages back to a Web Form.

DAO and RDO databinding components also cannot be upgraded. DAO and RDO code works as it did, but if you are designing applications with databinding, you should use ADO. This is supported in Visual Basic .NET.

OLE, shape controls & diagonal lines are examples of new ways to do the same thing, but it also fails during automatic upgrades. For example, the Web browser control allows you to embed Excel spreadsheets and other latebound documents. Although there is no automatic upgrade, the GDI+ methods have incredible functionality for drawing lines, shapes and pictures on your form, including alpha blending of pictures and loading and saving in many formats.

Property Pages - The Visual Basic 6 property browser was limited in the types it supported, so developers created custom property pages for editing properties such as image collections. These were a hack, often the most important properties were hidden in these pages. Visual Basic .NET has a richer property browser with built-in support for collections and multi-parameter properties. Also, you can define your own property editors for custom classes, so you can categorize and display everything in the property browser. Property pages are no longer needed and there is no upgrade for them.

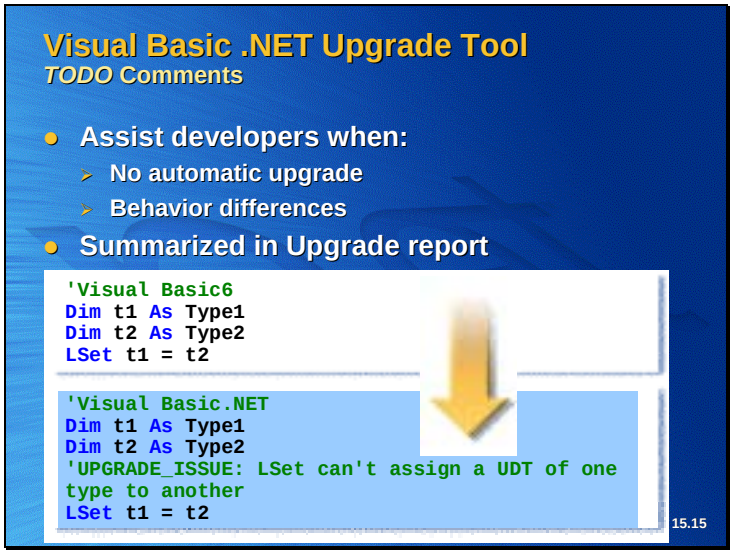
 ©2002 Microsoft Corporation

Add-in model for extensibility is different in Visual Studio .NET because it is now extended to cover every language. For example, you can write an add-in that adds code comments to both Visual C# and Visual Basic, but code that uses the VB6 Object model is not upgraded to the new Object model.

LSet, ObjPtr, VarPtr, StrPtr ObjPtr, VarPtr and StrPtr were undocumented functions for returning the memory address of variables. These were useful for performing a variety of hacks in Visual Basic 6 that are irrelevant in Visual Basic .NET. People are working on a white paper of replacement functions in .NET, so I won't talk about it any further here. LSet is similar in that it could be used to fit a type into a different type. You can do the same thing in Visual Basic .NET, but you will have your own method.

GoSub & Return, Computed GoTo: GoSub/Return has no equivalent in Visual Basic .NET, so avoid using GoSub/Return.

If you have applications that use these technologies, you can upgrade them to VB .NET if you rewrite some of the code. For example, ActiveX documents are added to your VB .NET project as related files, and ObjPtr will cause a compile error in Visual Basic .NET. The simple rule is to avoid these technologies in your Visual Basic 6 project.



Visual Basic .NET Upgrade Tool

TODO Comments

Upgrade tasks are explicit since each task and item in the report is associated with an online Help topic answering questions regarding the why's and how's behind code changes.

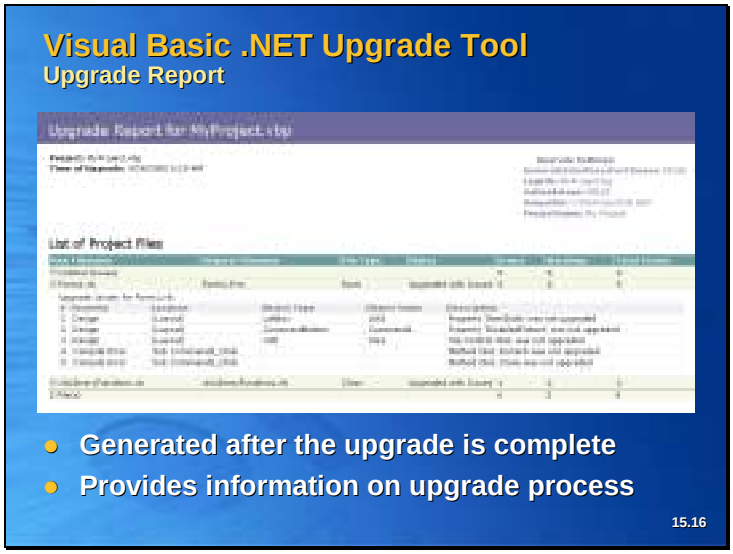
UPGRADE_ISSUE
Compile errors (must be fixed before code can compile)

UPGRADE_WARNING
Differences in behavior (should be examined before running code)

UPGRADE_TODO
Code partially upgraded (requires additional manual upgrade)

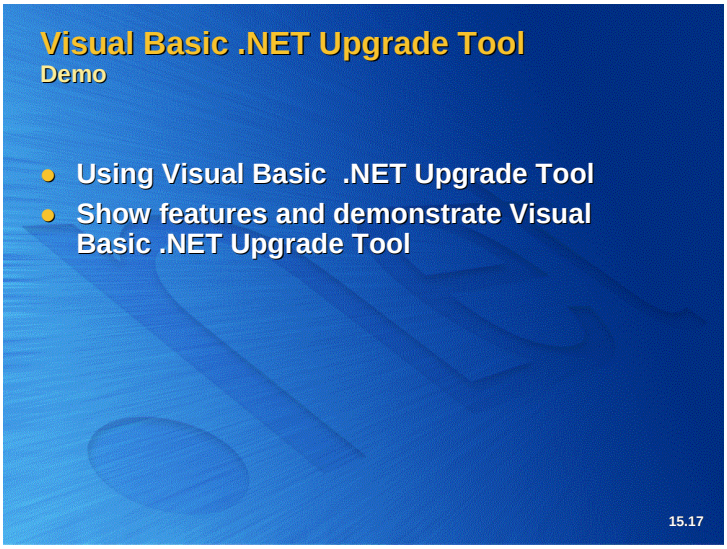
UPGRADE_NOTE
Code was significantly changed (no manual work necessary)

 ©2002 Microsoft Corporation



Visual Basic .NET Upgrade Tool

Upgrade Report


Visual Basic .NET Upgrade Tool
Demo
Using Visual Basic .NET Upgrade Tool
Show features and demonstrate Visual Basic .NET Upgrade Tool
15.17

Visual Basic .NET Upgrade Tool

Demo

Show the best methods and preparation for upgrading current VB 6 applications to .NET

How:

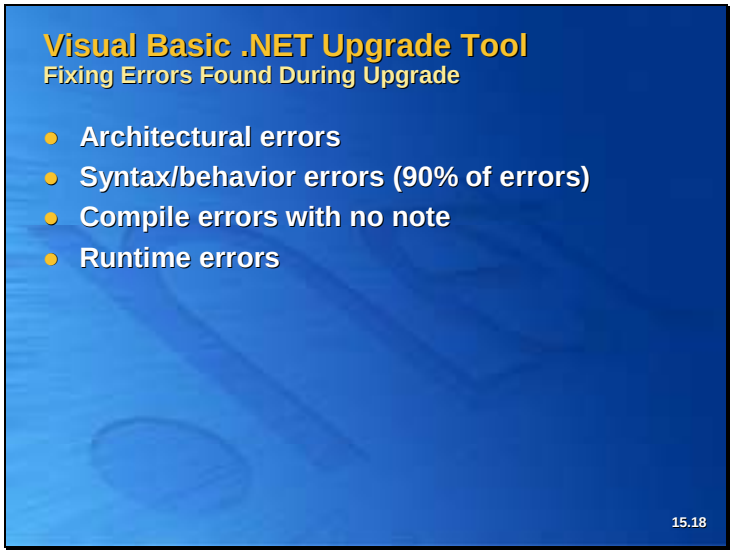
Two examples are reviewed:

1. A simple correct upgrade
2. A poorly planned and executed upgrade.

Show how both applications are upgraded

Discuss various issues with upgrading applications

 ©2002 Microsoft Corporation



Visual Basic .NET Upgrade Tool

Fixing Errors Found During Upgrade

After the upgrade process, errors may occur. You need to walk through these errors and fix them.

1. If the errors are architectural errors, then you need to change the component architecturally. For example:
   - **Painting to a form or picture box** - In Visual Basic 6, you can turn AutoRedraw on and then paint to the form/picture box. In Visual Basic .NET, move all that stuff into a paint event. (see the sample: VCR upgrade application)
   - **Printform**
   - **Forms collection**
2. Ninety percent of errors are syntax/behavior errors. The upgrade tool will report all errors with appropriate links to help. The different errors/ messages that provide information after a migration include:
   - **Upgrade_ISSUE**: These are compile errors that are not listed in the task list and must be fixed to successfully compile the code.
   - **Warning**: These are messages generated by the tool and are present in the task list. It is a good idea to look at these warning messages as ignoring them might cause problems later.
   - **TODO**: Any error in the code that needs to be fixed later is marked with a TODO comment. These issues then appear as part of the task list, and can be used later to locate any pending issues that need to be resolved.
   - A big change is Defiant, Defter
   - **'COULDN"T RESOLVE DEFAULT PROPERTY"** are commonly generated errors, because there are no default properties recognized by .NET. For example., consider the code below:

     ```
     dim v
     ```

```
v = 3
```

In this case, if v is a textbox, then Visual Basic 6.0 assumes that v = 3, implies v.text = 3, since text is the default property of a text box. Visual Basic .NET on the other hand will not be able to understand the same and it is difficult to understand what was intended here. The best solution is to hard code the Visual Basic code in the first place so that there are no compatibility issues.

3. Compile errors with no upgrade note:

```
Dim a()
ReDim a(4,3)
```

These errors are easy to fix, and generate a compile error. Runtime errors give no warning. You must use debugging skills to isolate the problem.

For example:

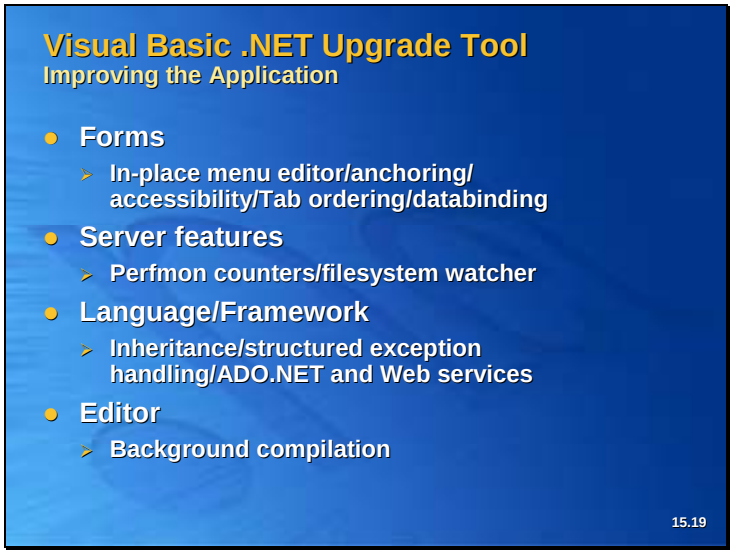
```
Treeview.Nodes.Add MyNode.Index,,"New Key"
```

MyNode.Index is parent for node. In Visual Basic .NET, you will get a runtime exception.

Isolate the problem to find error:

```
Dim n as integer
N = MyNode.Index
Treeview.Nodes.Add n,,"New Key"
```

 ©2002 Microsoft Corporation

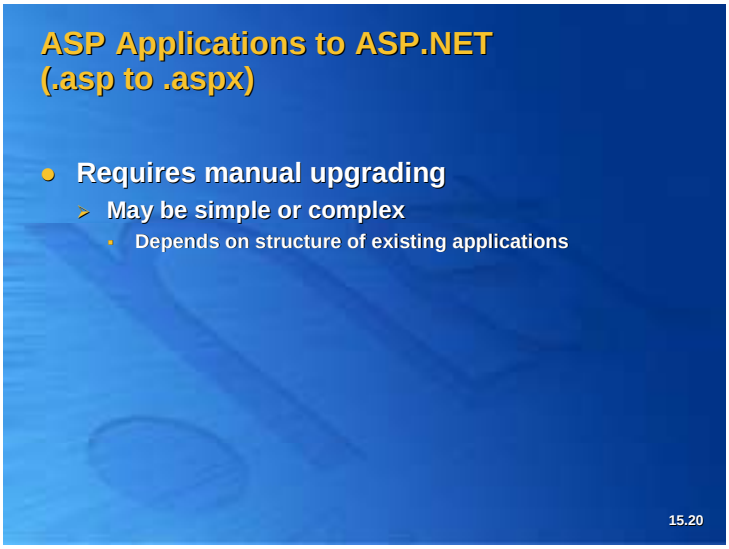


Visual Basic .NET Upgrade Tool

Improving the Application

If there are two or three method calls, isolate it to one call and find the problem:

- Understand the issues you will encounter
- COM Interop differences
- Resource issues – File handles open/db connections/resource handles left open
- Event orders in forms
- New .NET Windows Forms package – It doesn't reuse Visual Basic 6 forms code.



ASP Applications to ASP.NET

(.asp to .aspx)

Compatibility with few upgrades

Although ASP.NET is an overhaul of ASP, the two systems are compatible. There do exist, however, important components to consider before undertaking the effort of moving your Web Application from ASP to ASP.NET.

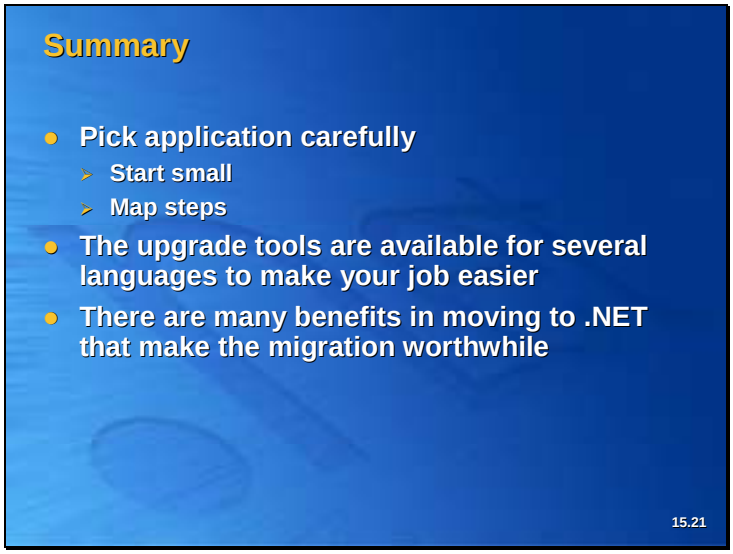
The majority of ASP applications use Visual Basic Scripting (Visual Basic Script) so most users should elect to migrate to ASP.NET using Visual Basic .NET. While not a requirement, changing languages at the same time you decide to port demands additional effort and is likely to include design and architectural changes. Because ASP and ASP.NET use separate file extensions (.asp vs. .aspx), separate configuration models (metabase/registry vs. XML-based configuration files) and separate processing engines, the two applications can operate side-by-side without adverse affects.

It also is possible for part of one application to use ASP and another part of the same application to use ASP.NET. Employing both systems simultaneously benefits those eager to move a large, rapidly changing site to ASP.NET one piece at a time.

Structural changes

Changes such as code blocks, programming languages, new page directives and render functions will affect the layout and coding style of Active Server Pages.

 ©2002 Microsoft Corporation


Summary
Pick application carefully
Start small
Map steps
The upgrade tools are available for several languages to make your job easier
There are many benefits in moving to .NET that make the migration worthwhile
15.21

Summary

There are many benefits in moving to .NET:

- Full and direct access to the .NET Framework
- Inheritance
- Language interoperability
- No DLL Conflict
- Improved RAD experience
- RAD for Windows, Web and server
- More Power
- No 'Visual Basic 6 glass ceiling'

Resources
Upgrading Visual Basic 6.0 Applications to Visual Basic .NET. The basics in upgrading.
A good starting place to get recommendations for upgrading your Visual Basic 6 Applications to Visual Basic .NET
http://msdn.microsoft.com/vstudio/techinfo/articles/upgrade/vbupgrade.asp
Preparing Your Visual Basic 6.0 Applications for the Upgrade to Visual Basic .NET
A developer's guide to getting ready for migrating Visual Basic 6 applications to .NET
http://msdn.microsoft.com
http://msdn.microsoft.com/vbasic/techinfo/articles/upgrade/default.asp
15.22

Resources

Upgrading Visual Basic 6.0 Applications to Visual Basic .NET. The basics in upgrading.

A good starting place to get recommendations for upgrading your Visual Basic 6 Applications to Visual Basic .NET:

http://msdn.microsoft.com/vstudio/techinfo/articles/upgrade/vbupgrade.asp

Preparing Your Visual Basic 6.0 Applications for the Upgrade to Visual Basic .NET.

A developer's guide to getting ready for migrating Visual Basic 6 applications to .NET:

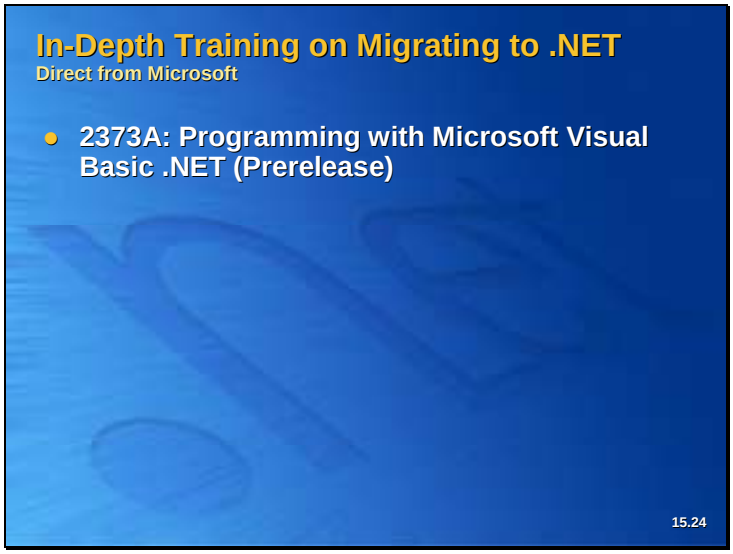
http://msdn.microsoft.com/

http://msdn.microsoft.com/vbasic/techinfo/articles/upgrade/default.asp

 ©2002 Microsoft Corporation

Resources

Migrating to ASP.NET: Key Considerations:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/Dnaspp/html/aspnetmigrissues.asp?frame=true


In-Depth Training on Migrating to .NET
Direct from Microsoft
2373A: Programming with Microsoft Visual Basic .NET (Prerelease)
15.24

In-Depth Training on Migrating to .NET

Direct from Microsoft

 ©2002 Microsoft Corporation

Microsoft
15.25